PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Peter D. Geiger; Manual J. Alvarez II; Thomas A. Dye
Assignee:	Quickshift, Inc. (f/k/a Interactive Silicon, Inc.)
Title:	System And Method For Managing Compression And Decompression Of System Memory In A Computer System

Serial No.:	09/915,751	Filing Date:	7/26/01
Examiner:	Kevin Verbrugge	Group Art Unit:	2188
Docket No.:	40532-P009US (f/k/a 5143-02501)		

Dallas, Texas
June 1, 2004

Mail Stop Appeal Brief - Patents
COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 C.F.R. § 1.191

Dear Sir:

Appellants submit this Appeal Brief pursuant to the Notice of Appeal filed in this case on December 23, 2003 and received in the Patent Office on or about December 29, 2003, having a shortened statutory period expiring on February 29, 2004, extended to June 1, 2004 (first business day after Saturday, May 29, 2004) by the petition and fee filed concurrently herewith. Also, enclosed is a check in the amount of $165.00, being the amount specified in 37 C.F.R. 1.17(c) for this Appeal Brief. The Commissioner is also authorized to deduct any other amounts required for this Appeal Brief and to credit any amounts overpaid to Deposit Account No. 23-2426. **This Brief is submitted in triplicate.**

I. REAL PARTY IN INTEREST

The real party in interest is the assignee, Quickshift, Inc., as named in the caption above. Please note that on February 19, 2004 Appellants filed for recording documentation which reflects that Quickshift, Inc. is the current assignee.

II. RELATED APPEALS AND INTERFERENCES

Based on information and belief, there are no appeals or interferences that could directly affect or be directly affected by or have a bearing on the decision by the Board of Patent Appeals in the pending appeal. However, Quickshift has appeals pending in the following application Serial Numbers: 09/915,751; 10/044,786; and 09/239,659.

III. STATUS OF CLAIMS

Claims 1-134 are pending in the application.

Claims 50-54 are allowed.

Claims 1-9, 17-22, 25-28, 33-45, 55-76, 78-89, 93-117 and 125-134 are rejected.

Claims 10-16, 23, 24, 29-32, 46-49, 77, 90-92 and 118-124 are objected to.

IV. STATUS OF AMENDMENTS

The Appellants' April 8, 2003 response and claim amendments to the Office Action dated January 2, 2003 have been considered, but the Examiner indicated that they did not place the application in condition for allowance because the Appellants' arguments were deemed unpersuasive.

V. SUMMARY OF THE INVENTION

In a prior art virtual memory system, the non-volatile memory (e.g., hard disk) is used as a secondary memory to provide the appearance of a greater amount of system memory. In such a prior art virtual memory system, as system memory becomes full, least recently used (LRU) pages are

swapped to the hard disk. These pages can be swapped back to the system memory when needed. There has been a need in the prior art to provide a method of increasing the effective size of system memory without increasing actual physical memory through use of a secondary memory, and to allow processors and/or I/O masters of the system to address more system memory than physically exists.

The present invention comprises various embodiments of a system, such as a computer system that includes a Compressed Memory Management Unit (CMMU) and one or more compression/decompression engines. The CMMU may operate in conjunction with the one or more compression/decompression engines to allow a processor or I/O master to address more system memory than physically exists. The CMMU may increase the effective size of system memory by keeping the least recently used pages compressed, and the most recently or frequently used pages uncompressed in physical memory. The CMMU may also increase the effective speed of system memory by storing the least recently used pages in a compressed format in system memory.

VI. ISSUES

A. Are claims 1-4, 22, 33-34, 36, 38-42, 55-61, 63, 67, 73, 78-79, 93-94, 97-98, 100, 102, 125, 129, and 131-134 properly rejected under 35 U.S.C. § 102(b) as being anticipated by U.S. Patent No. 5,699,539 to Garber et al. ("Garber")?

B. Are claims 5-9, 17-21, 25-28, 35, 37, 43-45, 62, 64-66, 68-72, 74-76, 80-89, 95-96, 99, 101, 103-117, 126-128, and 130 properly rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5,699,539 to Garber et al. ("Garber")?

VII. GROUPING OF THE CLAIMS

Claims 1-4, 22, 33-34, 36, 38-42, 55-61, 63, 67, 73, 78-79, 93-94, 97-98, 100, 102, 125, 129, and 131-134 form a first group.

Claims 5-9, 17-21, 25-28, 35, 37, 43-45, 62, 64-66, 68-72, 74-76, 80-89, 95-96, 99, 101, 103-117, 126-128, and 130 form a second group.

Claims 10-16, 23-24, 29-32, 46-54, 77, 90-92, 118-124 form a third group.

The reasons for these groupings is based on the Examiner's rejections.

VIII. ARGUMENT

For the reasons discussed below, Appellants respectfully traverse the Examiner's claim rejections under §§ 102 and 103 in view of Garber.

As discussed in Appellants' application, prior art computing systems having virtual memory capability typically use non-volatile memory (*e.g.*, a hard disk) as a secondary memory to provide the appearance of a greater amount of system memory. *See* page 1, lines 27-30. Appellants' application further notes that it is desirable to provide a method of increasing the effective size of system memory without having to rely on non-system memory (*e.g.*, the hard disk) as secondary memory.

Garber also discusses prior art virtual memory systems that rely on both primary (system memory) and secondary (hard disk) memory. *See* Col. 2, line 31 to Col. 3, line 52. Garber discloses a virtual memory system which is more reliant on primary memory, but nonetheless continues to rely on secondary memory. In Garber's system, uncompressed pages in current use are stored in a work space in system memory. On the other hand, all pages swapped out of the work space are stored in Mapped Out Storage Space that includes: (1) a portion of the system memory, identified as the Compression Heap; and (2) the computer's secondary memory (*i.e.*, hard disk). *See* Col. 3, line 66 to Col. 4, line 10. Garber further teaches that "in the preferred embodiment, for a computer system with 8 Mbytes of RAM, the virtual memory address space will be 16 Mbytes and the portion 132 of secondary memory included in the mapped out storage space will be 8 Mbytes. Col. 6, lines 54-59.

Garber discloses that preferably few, if any, pages are swapped out to secondary memory. *See* Col. 3, line 66 to Col. 4, line 3. Nonetheless, as is clear from Garber, the virtual memory system taught therein requires that secondary memory be present. In particular, when insufficient space is available in the primary memory portion of the Compression Heap to store swapped out pages, such pages are stored in secondary memory. *See* Abstract.

Unlike Garber, Appellants' claimed invention does not rely on secondary memory. Appellants' claimed memory system does not increase the effective size of system memory by relying on secondary memory to store the least used pages. Instead, in Appellants' claimed memory system, the effective size of system memory is increased by storing the least used pages in a compressed format only in system memory. *See, e.g.*, page 19, lines 9-11.

Each of the claims in Appellants' application is directed to the management of compressed and uncompressed pages in physical memory, "wherein the physical memory comprises the system memory." *See, e.g.*, claims 55 and 61. None of Appellants' claims are directed to a system that relies on secondary memory. Accordingly, for at least these reasons, Appellants submit that all the pending claims are allowable over Garber.

With regard to the Examiner's claim rejections under § 103 in view of Garber, Appellants respectfully submit that the Examiner has not met his burden of factually supporting his position that the claim elements not taught by Garber would have been "obvious" or "well-known."

It is the Examiner's burden to factually support any prima facie conclusion of obviousness. The Examiner's duty may not be satisfied by engaging in impermissible hindsight; any conclusion of obviousness must be reached on the basis of facts gleaned from the prior art. *See* MPEP §§ 2141-2144.

In a recent decision from the United States Court of Appeals for the Federal Circuit, the Federal Circuit noted that when the patent examiner and Board "rely on what they assert to be general knowledge to negate patentability, that knowledge must be articulated and placed on the record." *In re Sang-Su Lee*, 277 F.3d 1338, 1345 (Fed. Cir. 2002). Specifically, the Federal Circuit noted that conclusory statements about what is "basic knowledge" or "common sense" by themselves do not adequately support a determination of unpatentability. *See Id.* at 1343-44. Thus, the Federal Circuit held that findings of obviousness based on "common knowledge" must be supported by documented evidence that such knowledge exists. *See Id.* at 1344-45.

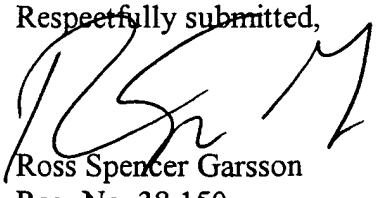
Here, the Examiner has only offered conclusory statements that the claim elements not taught by Garber would have been "obvious" or "well-known." The Examiner has not supported such statements with documented evidence, as he was required to do, especially after Appellants brought the lack of support to the Examiner's attention. Accordingly, the claims rejected under § 103 are allowable over Garber for at least this additional reason.

IX. CONCLUSION

For the above reasons, Appellants respectfully submit that any rejection of pending Claims 1-134 is unfounded. Accordingly, Appellants request that the rejection of any of Claims 1-134 be reversed.

This Brief is submitted in triplicate.

Respectfully submitted,



Ross Spencer Garsson
Reg. No. 38,150

CERTIFICATION UNDER 37 C.F.R. § 1.8

I hereby certify that this correspondence (along with any item referred to as being enclosed herewith) is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to Mail Stop Appeal Brief - Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on June 1, 2004.

Toni Stanley
Signature

TONI STANLEY

APPENDIX

1. A method for managing compression of pages of memory in a system comprising physical memory, wherein the physical memory comprises system memory, the method comprising:

receiving a system memory access;

locating a page translation entry for the system memory access in a page translation table;

determining if a page in the physical memory and referenced by the page translation entry is compressed or uncompressed;

if said determining indicates the page is compressed:

decompressing the compressed page to produce a decompressed page;

writing the decompressed page to the physical memory; and

providing a first physical memory address of the decompressed page in the physical memory to fulfill the system memory access.

2. The method of claim 1, further comprising:

if said determining indicates the page is uncompressed, providing a second physical memory address as indicated by the page translation entry to fulfill the system memory access.

3. The method of claim 1, further comprising:

if said determining indicates the page is uncompressed:

determining if the uncompressed page is to be compressed;

if said determining indicates the page is to be compressed:

compressing the uncompressed page to produce a compressed page; and

writing the compressed page to the physical memory.

4. The method of claim 3, wherein said compressing the uncompressed page comprises:

providing the uncompressed page to a compression engine; and

the compression engine compressing the page to produce the compressed page.

5. The method of claim 4, wherein said providing the page to the compression engine comprises:

a Direct Memory Access (DMA) channel reading the uncompressed page from the physical memory; and

the DMA channel writing the uncompressed page to the compression engine.

6. The method of claim 4, wherein said writing the compressed page to the physical memory comprises:

a Direct Memory Access (DMA) channel reading the compressed page from the compression engine; and

the DMA channel copying the compressed page into one or more linked compressed blocks in the physical memory.

7. The method of claim 6, further comprising locating the one or more linked compressed blocks for storing the compressed page in a list of available compressed blocks for storing compressed pages.

8. The method of claim 4, wherein said providing the page to the compression engine comprises:

a plurality of Direct Memory Access (DMA) channels reading the uncompressed page from the physical memory; and

the plurality of DMA channels writing the uncompressed page to the compression engine.

9. The method of claim 4, wherein said writing the compressed page to the physical memory comprises:

a plurality of Direct Memory Access (DMA) channels reading the compressed page from the compression engine; and

the plurality of DMA channels copying the compressed page into one or more linked compressed blocks in the physical memory.

10. The method of claim 3, wherein said compressing the uncompressed page comprises:

providing a different portion of the uncompressed page to each of a plurality of compression engines; and

the plurality of compression engines compressing the provided uncompressed portions of the page to produce compressed portions of the page.

11. The method of claim 10, wherein each of the plurality of compression engines implements a data compression algorithm, wherein the data compression algorithm is substantially the same for each of the plurality of compression engines.

12. The method of claim 10, wherein the plurality of compression engines compresses the uncompressed portions of the compressed page in parallel.

13. The method of claim 10, further comprising combining the compressed portions of the page to produce the compressed page.

14. The method of claim 3, wherein said compressing the uncompressed page comprises:

providing the uncompressed page to a plurality of compression engines, wherein each of the plurality of compression engines implements a different compression algorithm;

the plurality of compression engines each compressing the uncompressed page using the compression algorithm implemented by the particular compression engine to produce a plurality of compressed pages compressed by different compression algorithms;

selecting the compressed page from the plurality of compressed pages, wherein the selected compressed page has the highest compression ratio of the plurality of compressed pages.

15. The method of claim 14, further comprising marking the page translation entry associated with the selected compressed page to indicate the particular compression algorithm used in said compressing the page.

16. The method of claim 14, wherein the plurality of compression engines compresses the page in parallel.

17. The method of claim 1, wherein one or more recently used page translation entries from the page translation table are cached in a page translation cache, and wherein said locating a page translation entry comprises:

searching for the page translation entry associated with the system memory address in the page translation cache;

wherein, if said searching locates the page translation entry in the page translation cache, the page translation entry from the page translation cache is used in said determining if the page is compressed or uncompressed.

18. The method of claim 17, wherein, if the page translation entry is not located in the page translation cache, the method further comprises searching for the page translation entry in the page translation table;

wherein, if said searching locates the page translation entry in the page translation table, the page translation entry from the page translation table is used in said determining if the page is compressed or uncompressed.

19. The method of claim 18, further comprising caching the page translation entry located in the page translation table to the page translation cache as a recently used page translation entry.

20. The method of claim 17, wherein the page translation cache comprises a plurality of page translation entries, and wherein the page translation cache is fully associative.

21. The method of claim 20, wherein said searching for the page translation entry associated with the system memory address in the page translation cache comprises comparing the system memory address with all page translation entries in the page translation cache in parallel.

22. The method of claim 1, wherein said decompressing the page comprises:

providing the page to a decompression engine; and

the decompression engine decompressing the page to produce the decompressed page.

23. The method of claim 22, further comprising, if said determining indicates the page is compressed:

prior to said providing the page to a decompression engine:

examining the page translation entry to determine a compression algorithm used to compress the page; and

selecting the decompression engine from a plurality of decompression engines, wherein the decompression engine is configured to decompress data compressed using the determined compression algorithm.

24. The method of claim 22, wherein the decompression engine is a parallel decompression engine, wherein, in said decompressing the compressed page, the parallel decompression engine decompresses portions of the page in parallel.

25. The method of claim 22, wherein the page comprises one or more compressed blocks, and wherein said providing the page to the decompression engine comprises:

a Direct Memory Access (DMA) channel reading the one or more compressed blocks; and

the DMA channel copying the one or more compressed blocks to the decompression engine.

26. The method of claim 25, wherein the DMA channel reading the one or more compressed blocks comprises:

loading a physical memory address of a first compressed block into the DMA channel;

reading the first compressed block into the DMA channel; and

reading one or more subsequent compressed blocks linked to the first compressed block into the DMA channel.

27. The method of claim 25, wherein said writing the decompressed page to the physical memory is performed by the DMA channel, and wherein said writing the decompressed page to the physical memory comprises the DMA channel reading the uncompressed page from the decompression engine.

28. The method of claim 22, wherein the page comprises one or more compressed blocks, and wherein said providing the page to the decompression engine comprises:

a plurality of Direct Memory Access (DMA) channels reading the one or more compressed blocks; and

the plurality of DMA channels copying the one or more compressed blocks to the decompression engine.

29. The method of claim 1, further comprising:

providing a different portion of the compressed page to each of a plurality of decompression engines; and

each of the plurality of decompression engines decompressing the portion of the compressed page provided to the particular decompression engine.

30. The method of claim 29, wherein each of the plurality of decompression engines implements a data decompression algorithm, wherein the data decompression algorithm is substantially the same for each of the plurality of decompression engines.

31. The method of claim 29, wherein the plurality of decompression engines decompresses the portions of the compressed page in parallel.

32. The method of claim 29, further comprising combining the decompressed portions of the page to produce the decompressed page.

33. The method of claim 1, wherein said writing the decompressed page to the physical memory comprises:

locating a currently unused page in the physical memory in a list of currently unused pages for receiving uncompressed pages; and

writing the decompressed page as an uncompressed page to the located currently unused page.

34. The method of claim 1, wherein the compression of pages of the memory in the system is operable to increase the effective size of the system memory by keeping least recently used data as compressed data in the physical memory and most recently and frequently used data as uncompressed data in the physical memory.

35. The method of claim 34, wherein the system further comprises an operating system, wherein the operating system is aware of the increased effective size of the system memory.

36. The method of claim 34, wherein the system further comprises an operating system, wherein the operating system is not aware of the increased effective size of the system memory.

37. The method of claim 34, wherein the system further comprises an operating system, wherein the operating system is aware of the increased effective size of a first portion of the system memory, and wherein the operating system is not aware of the increased effective size of a second portion of the system memory.

38. The method of claim 1, wherein said decompressing the compressed page to produce the decompressed page comprises:

examining the page translation entry to determine if the page translation entry indicates the page is highly compressed, wherein, during compression of the page to generate the highly compressed page, the physical memory occupied by the page is freed for use by one or more processes executing within the system; and

if said examining determines the page is highly compressed:

allocating a portion of the physical memory for the page; and

writing data stored prior to said compressing the page to the page.

39. The method of claim 38, wherein the page translation entry includes a highly compressed attribute field, wherein the highly compressed attribute field indicates if the page is highly compressed, wherein said examining the page translation entry to determine if the page is highly compressed comprises examining the highly compressed attribute field.

40. The method of claim 39, wherein the highly compressed attribute field is a one-bit field.

41. A method for managing compression of pages of memory in a system comprising an operating system and physical memory, wherein the physical memory comprises system memory, the method comprising:

locating a page translation entry in a page translation table, wherein the page translation entry references an uncompressed page in the physical memory;

determining if the uncompressed page is to be compressed;

if said determining indicates the page is to be compressed:

compressing the uncompressed page to produce a compressed page; and

writing the compressed page to the physical memory;

wherein the compression of pages of the memory in the system is operable to increase the effective size of the system memory by keeping least recently used data as compressed data in the physical memory and most recently and frequently used data as uncompressed data in the physical memory; and

wherein the operating system is not aware of the increased effective size of the system memory.

42. The method of claim 41, wherein said compressing the uncompressed page comprises:

providing the uncompressed page to a compression engine; and

the compression engine compressing the page to produce the compressed page.

43. The method of claim 42, wherein said providing the page to the compression engine comprises:

one or more Direct Memory Access (DMA) channels reading the uncompressed page from the physical memory; and

the one or more DMA channels writing the uncompressed page to the compression engine.

44. The method of claim 42, wherein said writing the compressed page to the physical memory comprises:

one or more Direct Memory Access (DMA) channels reading the compressed page from the compression engine; and

the one or more DMA channels copying the compressed page into one or more linked compressed blocks in the physical memory.

45. The method of claim 44, further comprising locating the one or more linked compressed blocks for storing the compressed page in a list of available compressed blocks for storing compressed pages.

46. The method of claim 41, wherein said compressing the uncompressed page comprises:

providing a different portion of the uncompressed page to each of a plurality of compression engines; and

the plurality of compression engines each compressing the portion of the page which the particular compression engine was provided; and

combining the compressed portions of the page to produce the compressed page.

47. The method of claim 46, wherein the plurality of compression engines compress the portions of the compressed page in parallel.

48. The method of claim 41, wherein said compressing the uncompressed page comprises:

providing the uncompressed page to a plurality of compression engines, wherein each of the plurality of compression engines implements a different compression algorithm;

the plurality of compression engines each compressing the uncompressed page using the compression algorithm implemented by the particular compression engine to produce a plurality of compressed pages each compressed by a different compression algorithm;

selecting the compressed page from the plurality of compressed pages, wherein the selected compressed page has the highest compression ratio of the plurality of compressed pages.

49. The method of claim 48, further comprising marking the page translation entry associated with the compressed page to indicate the particular compression algorithm used in said compressing the page.

55. A method comprising:

determining a compression ratio for system memory in a system comprising physical memory, wherein the physical memory comprises the system memory;

determining if the compression ratio is below a compression ratio threshold;

if the compression ratio is below the compression ratio threshold:

locating a page translation entry in a page translation table referencing an uncompressed page in the system memory to be highly compressed;

setting a highly compressed attribute in the page translation entry to indicate that the page is highly compressed; and

freeing a first portion of the physical memory allocated to the page in the system memory;

wherein highly compressing the page of the memory in the system is operable to increase the compression ratio for the system memory.

56. The method of claim 55, wherein the highly compressed attribute is a one-bit field in the page translation entry.

57. The method of claim 55, further comprising:

receiving a system memory access for the page, wherein the system memory access requires that the page be uncompressed;

locating the page translation entry in the page translation table referencing the page;

determining that the page is highly compressed; and

allocating a second portion of the physical memory for the page.

58. The method of claim 57, further comprising filling the page with zeros after said allocating.

59. The method of claim 57, further comprising, after said allocating:

reading data from non-volatile storage; and

writing the data to the page.

60. The method of claim 55, further comprising, prior to said freeing the physical memory allocated to the page:

determining if the page is dirty; and

if the page is dirty, writing data from the page to non-volatile storage to make the page clean.

61. A system comprising:

one or more processors;

a physical memory comprising a system memory;

a system memory controller; and

a compressed memory management unit (CMMU), configured to:

receive from a first processor of the one or more processors a system memory access comprising a system memory address;

translate the system memory address into a first physical memory address;

cause the decompression of the compressed data at the first physical memory address in the physical memory;

write the decompressed data to a second physical memory address; and

pass the second physical memory address to the system memory controller;

wherein the system memory controller is configured to fulfill the system memory access from the decompressed data at the second physical memory address; and

wherein the CMMU is operable to increase the effective size of the system memory by keeping least recently used data as compressed data in the physical memory and most recently and frequently used data as uncompressed data in the physical memory.

62. The system of claim 61, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is aware of the increased effective size of the system memory.

63. The system of claim 61, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is not aware of the increased effective size of the system memory.

64. The system of claim 61, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is aware of the increased effective size of a first portion of the system memory, and wherein the operating system is not aware of the increased effective size of a second portion of the system memory.

65. The system of claim 61, wherein the CMMU comprises:

a page translation cache configured to cache page translation entries; and

one or more scatter/gather Direct Memory Access (DMA) channels configured for transferring data from the CMMU to one or more destinations and for receiving data on the CMMU from one or more sources.

66. The system of claim 1765, wherein the CMMU further comprises a compression/decompression engine configured to compress uncompressed data and to decompress compressed data under control of the CMMU.

67. The system of claim 61, further comprising:

a page translation table comprising one or more page translation entries;

wherein, in said translating the system memory address into a first physical memory address;

the CMMU is further configured to:

locate a page translation entry for the system memory address in the page translation table; and

determine the first physical memory address from the page translation entry for the system memory address.

68. The system of claim 67, wherein the CMMU further comprises:

a page translation cache comprising one or more cached page translation entries from the page translation table;

wherein, in said locating a page translation entry, the CMMU is further configured to search for the page translation entry associated with the system memory address in the page translation cache;

wherein, if said searching the page translation cache locates the page translation entry in the page translation cache, the page translation entry from the page translation cache is used in said determining the first physical memory address.

69. The system of claim 68, wherein, if said searching the page translation cache does not locate the page translation entry in the page translation cache, the CMMU is further configured to:

search for the page translation entry in the page translation table;

wherein, if said searching the page translation table locates the page translation entry in the page translation table, the page translation entry from the page translation table is used in said determining the first physical memory address.

70. The system of claim 69, wherein the CMMU is further configured to cache the page translation entry located in the page translation table to the page translation cache as a recently used page translation entry.

71. The system of claim 68, wherein the page translation cache is fully associative.

72. The system of claim 71, wherein, in said searching for the page translation entry associated with the system memory address in the page translation cache, the CMMU is further configured to compare the system memory address with all page translation entries in the page translation cache in parallel.

73. The system of claim 61, wherein the system further comprises:

a compression/decompression engine;

wherein, in said causing the decompression of the compressed data, the CMMU is further configured to write the compressed data to the compression/decompression engine; and

wherein the compression/decompression engine is configured to decompress the compressed data to produce the decompressed data.

74. The system of claim 73, wherein the CMMU further comprises:

one or more Direct Memory Access (DMA) channels;

wherein the compressed data comprises one or more compressed blocks, and wherein, in said writing the compressed data to the compression/decompression engine, the one or more DMA channels are configured to:

read the one or more compressed blocks; and

copy the one or more compressed blocks to the compression/decompression engine.

75. The system of claim 74, wherein the CMMU is further configured to:

load a physical memory address of a first compressed block into the one or more DMA channels; and

wherein in reading the one or more compressed blocks, the one or more DMA channels are further configured to read the first compressed block and one or more subsequent compressed blocks linked to the first compressed block.

76. The system of claim 74, wherein said writing the decompressed data to the second physical memory address is performed by the one or more DMA channels, and wherein, in said writing the decompressed data to the second physical memory address, the one or more DMA

channels are further configured to read the decompressed data from the compression/decompression engine.

77. The system of claim 61, wherein the system further comprises a plurality of compression/decompression engines, wherein at least two of the plurality of compression/decompression engines implement different compression/decompression algorithms, and wherein the CMMU is further configured to:

prior to said writing the compressed data to the compression/decompression engine:

determine a particular compression algorithm used to compress the compressed data; and

select the compression/decompression engine from the plurality of compression/decompression engines, wherein the selected compression/decompression engine is configured to decompress data compressed using the particular compression algorithm.

78. The system of claim 61, wherein the system further comprises:

a compression/decompression engine configured to decompress compressed data under control of the CMMU.

79. The system of claim 78, wherein the compression/decompression engine is a parallel compression/decompression engine configured to perform parallel data decompression under control of the CMMU.

80. The system of claim 78, wherein the compression/decompression engine is comprised in the CMMU.

81. The system of claim 61, wherein the CMMU is comprised in one of the one or more processors.

82. The system of claim 81, wherein at least one of the one or more processors further comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

83. The system of claim 81, wherein the system memory controller comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

84. The system of claim 81, wherein the physical memory comprises one or more memory modules, and wherein at least one of the one or more memory modules comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

85. The system of claim 61, wherein the CMMU is comprised in the system memory controller.

86. The system of claim 85, wherein the system memory controller further comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

87. The system of claim 85, wherein the physical memory comprises one or more memory modules, and wherein at least one of the one or more memory modules comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

88. The system of claim 61, wherein the physical memory comprises one or more memory modules, wherein the CMMU is coupled to the system memory controller and the one or more memory modules.

89. The system of claim 88, wherein at least one of the one or more memory modules comprises a compression/decompression engine configured to decompress compressed data under control of the CMMU.

90. The system of claim 61, wherein the system further comprises:

a plurality of compression/decompression engines;

wherein, in said causing the decompression of the compressed data, the CMMU is further configured to write a different portion of the compressed data to each of the plurality of compression/decompression engines;

wherein the plurality of compression/decompression engines are configured to decompress the portions of the compressed data to produce a plurality of decompressed data portions; and

wherein the CMMU is further configured to combine the plurality of decompressed data portions to produce the decompressed data.

91. The system of claim 90, wherein each of the plurality of compression/decompression engines implements a data decompression algorithm, wherein the data decompression algorithm is substantially the same for each of the plurality of compression/decompression engines.

92. The system of claim 90, wherein the plurality of compression/decompression engines decompress the portions of the compressed data in parallel.

93. The system of claim 61, wherein, in said writing the decompressed data to physical memory, the CMMU is further configured to:

locate a currently unused page in the physical memory in a list of currently unused pages for receiving uncompressed data; and

write the decompressed data as an uncompressed page to the located currently unused page.

94. The system of claim 61, wherein the CMMU is further configured to manage the system memory on a page granularity.

95. The system of claim 61, wherein page size is programmable.

96. The system of claim 61, wherein a maximum compression ratio applied by the CMMU is programmable.

97. The system of claim 61, further comprising a kernel driver configured to:

monitor an actual compression ratio achieved by the CMMU; and

ensure that a minimum compression ratio is maintained by the CMMU.

98. A system comprising:

one or more processors;

a system memory controller;

a physical memory comprising a system memory; and

a compressed memory management unit (CMMU), configured to:

translate a system memory address into a first physical memory address;

cause the compression of the uncompressed data at the first physical memory address to produce compressed data; and

write the compressed data to a second physical memory address;

wherein the system is operable to increase the effective size of system memory by keeping least recently used pages compressed in the physical memory and most recently and frequently used pages uncompressed in the physical memory.

99. The system of claim 98, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is aware of the increased effective size of the system memory.

100. The system of claim 98, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is not aware of the increased effective size of the system memory.

101. The system of claim 98, wherein the system further comprises program instructions executable within the system to implement an operating system, wherein the operating system is aware of the increased effective size of a first portion of the system memory, and wherein the operating system is not aware of the increased effective size of a second portion of the system memory.

102. The system of claim 98, further comprising:

wherein the CMMU is further configured to receive from a first processor of the one or more processors the system memory access comprising a system memory address.

103. The system of claim 98, wherein the CMMU comprises:

a page translation cache configured to cache page translation entries; and

one or more scatter/gather Direct Memory Access (DMA) channels configured for transferring data from the CMMU to one or more destinations and for receiving data on the CMMU from one or more sources.

104. The system of claim 103, wherein the CMMU further comprises a compression/decompression engine configured to compress uncompressed data and to decompress compressed data under control of the CMMU.

105. The system of claim 103, wherein the page translation cache is fully associative.

106. The system of claim 98, wherein the system further comprises:

a COMPRESSION/decompression engine configured to compress uncompressed data under control of the CMMU.

107. The system of claim 106, wherein the compression/decompression engine is a parallel compression/decompression engine configured to perform parallel data compression under control of the CMMU.

108. The system of claim 106, wherein the compression/decompression engine is comprised in the CMMU.

109. The system of claim 98, wherein the CMMU is comprised in one of the one or more processors.

110. The system of claim 109, wherein at least one of the one or more processors comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

111. The system of claim 109, wherein the system memory controller comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

112. The system of claim 109, wherein the physical memory comprises one or more memory modules, and wherein the at least one of the one or more memory modules comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

113. The system of claim 98, wherein the CMMU is comprised in the system memory controller.

114. The system of claim 113, wherein the system memory controller further comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

115. The system of claim 113, wherein the physical memory comprises one or more memory modules, and wherein at least one of the one or more memory modules comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

116. The system of claim 98, wherein the physical memory comprises one or more memory modules, wherein the CMMU is coupled to the system memory controller and the one or more memory modules.

117. The system of claim 116, wherein at least one of the one or more memory modules comprises a compression/decompression engine configured to compress uncompressed data under control of the CMMU.

118. The system of claim 98, wherein the system further comprises:

a plurality of compression/decompression engines;

wherein, in said causing the compression of the uncompressed data, the CMMU is further configured to write a different portion of the uncompressed data to each of the plurality of compression/decompression engines;

wherein the plurality of compression/decompression engines are configured to compress the portions of the uncompressed data to produce a plurality of compressed data portions; and

wherein the CMMU is further configured to combine the plurality of compressed data portions to produce the compressed data.

119. The system of claim 118, wherein each of the plurality of compression/decompression engines implements a data compression algorithm, wherein the data compression algorithm is substantially the same for each of the plurality of compression/decompression engines.

120. The system of claim 118, wherein the plurality of compression/decompression engines is configured to compress the portions of the uncompressed data in parallel.

121. The system of claim 98, wherein the system further comprises:

a plurality of compression/decompression engines, wherein each of the plurality of compression/decompression engines implements a different compression algorithm;

wherein, in said causing the compression of the uncompressed data, the CMMU is further configured to provide the uncompressed data to each of the plurality of compression/decompression engines;

wherein the plurality of compression engines are configured to each compress the uncompressed data using the compression algorithm implemented by the particular compression engine to produce a plurality of compressed data each compressed by a different compression algorithm; and

wherein, in said causing the compression of the uncompressed data, the CMMU is further configured to select the compressed data from among the plurality of compressed data.

122. The system of claim 121, wherein the CMMU selects the compressed data with the highest compression ratio from among the plurality of compressed data.

123. The system of claim 121, wherein the system further comprises a page translation table comprising one or more page translation entries, wherein one of the one or more page translation entries references a page of physical memory at the second physical memory address, wherein the CMMU is further configured to mark the page translation entry to indicate the particular compression algorithm used in said compressing the uncompressed data.

124. The system of claim 121, wherein the plurality of compression/decompression engines is configured to compress the uncompressed data in parallel.

125. The system of claim 98, further comprising:

a compression/decompression engine;

wherein, in said causing the compression of the uncompressed data, the CMMU is further configured to write the uncompressed data to the compression/decompression engine; and

wherein the compression engine is configured to compress the uncompressed data to produce the compressed data.

126. The system of claim 125, wherein the CMMU further comprises:

one or more Direct Memory Access (DMA) channels;

wherein, in said writing the uncompressed data to the compression/decompression engine, the one or more DMA channels are configured to:

read the uncompressed data from physical memory; and

write the uncompressed data to the compression/decompression engine.

127. The system of claim 125, wherein, in said writing the compressed data to the second physical memory address, the one or more DMA channels are further configured to:

read the compressed page from the compression/decompression engine; and

copy the compressed data into one or more linked compressed blocks in physical memory.

128. The system of claim 125, wherein the CMMU is further configured to:

locate the one or more compressed blocks for storing the compressed data in a list of available compressed blocks for storing compressed data; and

link the one or more compressed blocks to generate the one or more linked compressed blocks.

129. A system comprising:

one or more processors;

a system memory controller;

a physical memory comprising a system memory; and

a compressed memory management unit (CMMU), configured to:

maintain a threshold compression ratio, wherein the threshold compression ratio is a desired minimum ratio between compressed pages and uncompressed pages of the system memory;

dynamically monitor a current compression ratio for the system memory, wherein the current compression ratio is an actual ratio between compressed pages and uncompressed pages of the system memory, and wherein the compression ratio determines an amount by which system memory address space can be increased;

dynamically determine if the current compression ratio is below the threshold compression ratio;

if the current compression ratio is below the threshold compression ratio, cause the compression of one or more uncompressed pages to generate one or more compressed pages; and

wherein the CMMU is operable to increase the effective size of the system memory by keeping least recently used pages compressed in the physical memory and most recently and frequently used pages uncompressed in the physical memory; and

wherein compressing the one or more uncompressed page is operable to increase the current compression ratio for the system memory.

130. The system of claim 129, wherein the threshold compression ratio is programmable to determine an amount by which the effective size of the system memory can be increased.

131. The system of claim 129, further comprising:

a compression engine;

wherein, in said causing the compression of the one or more uncompressed pages, the CMMU is further configured to write the uncompressed pages to the compression engine; and

wherein the compression engine is configured to compress the one or more uncompressed pages to produce the one or more compressed pages.

132. The system of claim 129, wherein, in said causing the compression of the one or more uncompressed pages, the CMMU is further configured to:

locate a page translation entry in a page translation table referencing an uncompressed page in the system memory;

determine that the uncompressed page is highly compressible;

set a highly compressed attribute in the page translation entry to indicate that the page is highly compressed; and

free a portion of the physical memory allocated to the page in the system memory;

wherein highly compressing the uncompressed page is operable to increase the current compression ratio for the system memory.

133. The system of claim 132, wherein the highly compressed attribute is a one-bit field in the page translation entry.

134. The system of claim 129, wherein the CMMU is further configured to, prior to said freeing the portion of the physical memory allocated to the page:

determine if the page is dirty; and

if the page is dirty, write data from the page to non-volatile storage to make the page clean.